

Hope you are doing well & are ready for the examination.

Objectives

You as an Offensive Specialist have been tasked to perform red team redirector setup for traffic management and implant delivery during the red team engagement. Your tasks is to access the redirector machine (via SSH) and fix the broken configuration file as per the asked questions, test it and validate your answers as flags in the portal.

You must finish the examination **within 6 hours** with all the correct 10 answers in order to earn CRT-ID Certification.

Machine Access Details

Login IP	Username	Password
192.168.1.1	ubuntu	1qaz!@WSX

SSH to the machine using given password:

```
● ● ●
$ ssh ubuntu@192.168.1.1
```

NOTE

- You are free to create as many as SSH connection to the instance.
- The redirector machine is given to the students for testing and answering the flag based questions only.

Environment & Configuration:

- You will log into a pre-configured Debian machine using the above credentials
- The dummy payload is present in the "/opt/backend" directory

```
● ● ●
$ ls /opt/backend/payloads/implant.bat
```

- A broken Nginx configuration file is present at:

```
● ● ●
$ /etc/nginx/sites-available/broken.conf
```

- For "proxy_pass" functionality, the web server (at TCP 8080 Port) is already running at "/opt/backend", it serves the payload directory which contains "implant.bat" file. In case if the web server breaks, feel free to start another process at the same "/opt/backend" using:

```
● ● ●
$ cd /opt/backend
$ python3 -m http.server 8080
```

- You are required to identify and fix the configuration issues based on the individual questions provided to you. You are free to edit the broken file in any way to achieve the desired functionality as per the asked questions.

Commands for Verification & Debugging:

- Test syntax validity after "broken.conf" updation:

```
● ● ●
$ nginx -t
```

- Reload nginx service:

```
● ● ●
$ systemctl reload nginx
```

- View nginx error log:

```
● ● ●
$ tail -f /var/log/nginx/error.log
```

Testing Commands for validating your configuration file:

1. Basic Connectivity Test:

```
● ● ●  
$ curl -I http://localhost/  
$ curl http://localhost/status
```

2. IP Whitelisting Test (should work only for allowed IPs):

```
● ● ●  
$ curl -I http://localhost/download/file.update
```

3. User-Agent Filtering (Allow only CRT-ID):

```
● ● ●  
$ curl -A "CRT-ID" http://localhost/download/file.update
```

4. Extension Spoofing Detection (Only works with correct User Agent):

```
● ● ●  
$ curl -A "CRT-ID" -v http://localhost/download/file.update
```

5. Rate Limiting Test (burst-allowed requests will pass, then 503):

```
● ● ●  
$ for i in {1..7}; do  
    echo "Request $i:"  
    curl -A "CRT-ID" -I http://localhost/download/file.update  
    echo "---"  
done
```

6. Direct Payload Redirection (Google.com or LinkedIn redirect):

```
● ● ●  
$ curl -I http://localhost/payloads/implant.bat
```

Important Notes:

All answers must follow the Flag Format provided in each question.

Few answers require configuration file updation and testing as per the requirements & some can be answered without any testing

You are free to modify the configuration in any way to achieve all correct 10 questions in order to earn CWL CRT-ID certificate.

In total you will have 2 exam attempts, in case if you fail 1st, you will have a chance to attempt another. If you fail both, email support@cyberwarfare.live for extension.

1. Block All Non-CRT-ID User-Agents

Scenario: The red team's command tools use a unique User-Agent header ("CRT-ID") that must be allowed. All other tools, scanners, or browsers should be blocked with HTTP 403 to reduce detection.

Task: Write the complete if block that performs case-insensitive matching to allow only the "CRT-ID" User-Agent.

Flag Format: if (\$http_user_agent !~* "REGEX_PATTERN") { return STATUS_CODE; }

```
if ($http_user_agent !~* "CRT-ID") { return 403; }
```

Completed

2. Whitelist Only 10.10.10.0/24 for "/download"

Scenario: The payload endpoint "/download" must only be accessible to red team machines inside the 10.10.10.0/24 operational subnet. All others must be denied to prevent exposure.

Task: Write the allow/deny block where allow comes before deny, using "deny all;" as the deny rule.

Expected Answer Format: Two-line access control

Flag Format: allow 0.0.0.0/24;deny all;

```
allow 10.10.10.0/24;deny all;
```

Completed

3. Rewrite Specific File Only

Scenario: A red team beacon fetches the payload by requesting "/download/file.update". This must be internally mapped to "/payloads/implant.bat" without exposing the real path or triggering an external redirect.

Task: Write the exact rewrite directive (with regex) to internally map /download/file.update to /payloads/implant.bat using break.

Expected Answer Format: Complete Nginx rewrite directive

Flag Format: rewrite ^/path/to/file\$ /new/path break;

```
rewrite ^/download/file\.update$ /payloads/implant.bat break;
```

Completed

4. Add Redirection Logic for Direct Payload Access

Scenario: If anyone tries to directly access /payloads/implant.bat (such as blue team analysts), they should be redirected to a decoy destination to avoid revealing real infrastructure.

Task: Write the location block that redirects direct access to /payloads/implant.bat to https://www.google.com using 302 redirect.

Expected Answer Format: Full location block

Flag Format: location = /file { return 302 URL; }

```
location = /payloads/implant.bat { return 302 https://www.google.com; }
```

5. Rate Limit Burst Protection

Scenario: Your red team's legitimate automated tools are being blocked by the current rate limiting configuration. You need to allow a small burst of requests to accommodate normal operation patterns while still preventing blue team enumeration attempts.

Task: Write the complete "limit_req" directive that uses the ratelimit_zone, allows a burst of 2 additional requests, and applies the nodelay parameter for immediate processing.

Expected Answer Format: Single directive

Flag Format: limit_req zone=ZONE_NAME burst=X nodelay;

```
limit_req zone=ratelimit_zone burst=2 nodelay;
```

Completed

6. Change Internal Proxy Port

Scenario: Your team server hosting the actual payloads has been moved to a different port for operational security. The internal payload server is now running on port 8080 instead of port 8443.

Task: Write the complete proxy_pass directive that forwards requests to the internal server running on 127.0.0.1:8443.

Expected Answer Format: Complete proxy_pass directive with semicolon

Flag Format: proxy_pass http://IP:PORT/;

```
proxy_pass http://127.0.0.1:8080/;
```

7. Redirect to Business Site for OpSec

Scenario: For better operational security against blue team analysis, direct access to your payload should redirect to a commonly accessed business site. Change the redirect target to https://www.linkedin.com to blend in with corporate network traffic.

Task: Write the complete return directive that will perform a temporary redirect (302) to https://www.linkedin.com.

Expected Answer Format: Complete return directive with status code and URL

Flag Format: return 302 URL;

```
return 302 https://www.linkedin.com;
```

Completed

8. Create a Status Endpoint with IP Whitelist

Scenario: You need to expose a "/status" endpoint that allows local health checks by the redirector itself, but blocks all other external monitoring tools.

Task: Write a location block that returns a plain 200 "OK" when "/status" is accessed, but only if the request comes from 127.0.0.1.

Expected Answer Format: Full location block

Flag Format: location /path { allow IP; deny all; return CODE "RESPONSE"; }

```
location /status { allow 127.0.0.1; deny all; return 200 "OK"; }
```

Completed

9. Match User-Agent "CRT-ID123" or Similar with Digits

Scenario: Your team rotates User-Agent values dynamically by appending numbers, e.g., CRT-ID001, CRT-ID77, etc. You want to allow these variants only, while blocking all others.

Task: Write a case-insensitive regex to match user-agent "CRT-ID123" or "crt-id99" using an if block that allows access only if the pattern matches.

Expected Answer Format: Full if block

Flag Format: if (\$http_user_agent !~* "REGEX") { return 403; }

```
if ($http_user_agent !~* "^CRT-ID\d+$") { return 403; }
```

Completed

10. Block Default User Agent String

Scenario: You've noticed that most automated scans and basic recon tools (like curl) use a default User-Agent string that includes the word "curl". You want to proactively block all such requests to prevent unauthorized enumeration.

Task: Write the complete Nginx if block that returns 403 when the request's User-Agent contains the string "curl", case-insensitively.

Expected Answer Format: Full if block using a case-insensitive match

Flag Format: if (\$http_user_agent ~* "STRING") { return 403; }

```
if ($http_user_agent ~* "curl") { return 403; }
```

The config file is enough to answer all questions – copy paste only !

```
server {
listen 80;
server_name default;

# Block All Non-CRT-ID User-Agents & Match User-Agent with Digits
location /download {
# Block All Non-CRT-ID User-Agents
if ($http_user_agent !~* "CRT-ID") { return 403; }

# Match User-Agent "CRT-ID123" or Similar with Digits
if ($http_user_agent !~* "^CRT-ID\d+$") { return 403; }

# Block Default User Agent String
if ($http_user_agent ~* "curl") { return 403; }

# Whitelist Only 10.10.10.0/24 for "/download"
allow 10.10.10.0/24;
deny all;
```

```
# Rate Limit Burst Protection
limit_req zone=ratelimit_zone burst=2 nodelay;

# Rewrite Specific File Only
rewrite ^/download/file\.update$ /payloads/implant.bat break;

# Change Internal Proxy Port
proxy_pass http://127.0.0.1:8080/;
}

# Create a Status Endpoint with IP Whitelist
location /status {
allow 127.0.0.1;
deny all;
return 200 "OK";
}

# Redirect to Business Site for OpSec (Direct Payload Access)
location = /payloads/implant.bat {
return 302 https://www.linkedin.com;
}
}
```